

# A Globally Convergent Algorithm for the Run-to-Run Control of Systems with Sector Nonlinearities

Grégory François,<sup>†</sup> Bala Srinivasan,<sup>\*</sup> and Dominique Bonvin<sup>\*,†</sup>

Laboratoire d'Automatique École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland, and  
Department of Chemical Engineering École Polytechnique de Montreal, Montreal, Canada, H3C 3A7

Run-to-run control is a technique that exploits the repetitive nature of processes to iteratively adjust the inputs and drive the run-end outputs to their reference values. It can be used to control both static and finite-time dynamic systems. Although the run-end outputs of dynamic systems result from the integration of process dynamics during the run, the relationship between the input parameters  $p$  (fixed at the beginning of the run) and the run-end outputs  $z$  (available at the end of the run) can be seen as the static map  $z(p)$ . Run-to-run control consists in computing the input parameters  $p^*$  that lead to the reference values  $z_{\text{ref}}$ . Although a wide range of techniques have been reported, most of them do not guarantee global convergence, that is, convergence toward  $p^*$  for all possible initial conditions. This paper presents a new algorithm that guarantees global convergence for the run-to-run control of both static and finite-time dynamic systems. Attention is restricted to sector nonlinearities, for which it is shown that a fixed-gain update can lead to global convergence. Furthermore, since convergence can be very slow, it is proposed to take advantage of the mathematical similarity between run-to-run control and the solution of nonlinear equations, and combine the fixed-gain algorithm with a faster variable-gain quasi-Newton algorithm. Global convergence of this hybrid scheme is proven. The potential of this algorithm in the context of run-to-run optimization of dynamic systems is illustrated via the simulation of an industrial batch polymerization reactor.

## 1. Introduction

In the last 20 years, run-to-run control has developed as an important tool for process improvement in industry.<sup>1</sup> The basic idea is to exploit the repetitive nature of discontinuous processes in chemical production, mechanical machining, or semiconductor manufacturing to determine the subsequent inputs on the basis of previous run-end measurements.<sup>2</sup> In its earliest version, run-to-run control was mainly motivated by the lack of *in situ* measurements for the control of film thickness or electrical properties in semiconductors. Typically, these quantities could not be measured in real-time and used for online feedback control;<sup>3</sup> hence, there was the idea of considering process operation as a succession of runs and of using off-line measurements to adjust, on a run-to-run basis, the set points of online controllers.<sup>2,3</sup> The fact that run-to-run control requires only measurements that are available at the end of the run is one reason that explains its popularity in an industrial setting. Another reason is the fact that control can be used to improve process performance. Indeed, it is possible to optimize a process with the concept of necessary conditions of optimality (NCO) tracking.<sup>4</sup> This way, an optimization problem can be turned into a control problem, for which the control objective is to satisfy the NCO. Hence, run-to-run control is an efficient tool for optimizing repetitive finite-time dynamic processes.<sup>5,6</sup> The implementation of run-to-run control is greatly simplified if the input profiles can be parametrized. This way, the map between the manipulated and controlled variables can be seen, from a run-to-run perspective and upon integration of the process dynamics, as a static map.<sup>7</sup> Said differently, a completed run can be seen as the map  $z(p)$  between the parameters  $p$  that characterize the input trajectories and the outputs  $z$  measured at final time.

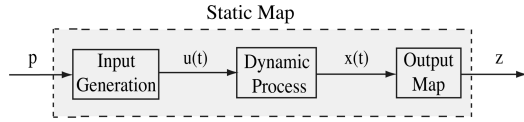
Regardless of its use for either bringing outputs to desired reference values (run-to-run control) or iteratively improving process performance (run-to-run optimization), convergence of a run-to-run scheme is a critical issue. The reformulation of the original dynamic control problem into a static control problem opens the way for the use of fixed-point theory or Lyapunov-type approaches to enforce stability.<sup>8</sup> Recently, tuning rules that guarantee global convergence to the desired references  $z_{\text{ref}}$  have been proposed for systems with sector nonlinearities.<sup>9,10</sup> Fixed-gain updates have been shown to converge to the global solution, provided the slope of the upper bounding sector is known and the gain of the update law is smaller than twice the inverse of this slope.<sup>8</sup> However, fixed-gain updates result in slow convergence, which is particularly critical in batch processing, since the slower the convergence, the more runs are needed to reach optimality.

In this paper, we propose to exploit the similarities between the two problems of run-to-run control, where the input parameters  $p$  are updated iteratively to bring the run-end outputs to the desired references  $z(p^*) = z_{\text{ref}}$ , and the problem of solving sets of nonlinear equations, for which the solution  $p^*$  to the set of equations  $z(p) = 0$  is sought iteratively. The solution of nonlinear equation systems has been widely studied in the literature,<sup>11,12</sup> and several numerical methods that exhibit fast convergence are available.<sup>11,13,14</sup> This work proposes to combine the fixed low-gain iterative scheme with a quasi-Newton type of update to benefit from the faster convergence of Newton-type algorithms. The gain matrix will be updated via the Jacobian matrix using for example Broyden's formula, except in the neighborhood of a local minimum, where a fixed low gain will be used. It will be shown that global convergence can be ensured despite switching between the two algorithms. The main advantage of this hybrid algorithm, compared to the fixed-gain algorithm, is its relative high convergence speed, while still preventing getting stuck in a local minimum.<sup>15</sup>

\* To whom correspondence should be addressed. E-mail: dominique.bonvin@epfl.ch.

<sup>†</sup> École Polytechnique Fédérale de Lausanne.

<sup>\*</sup> École Polytechnique de Montreal.



**Figure 1.** Static map between the input parameters  $p$  and the run-end outputs  $z$  of a finite-time dynamic process.

The paper is organized as follows. Section 2 presents the essence of run-to-run control. The concept of sector nonlinearity is introduced in section 3, together with the fixed-gain algorithm and its convergence analysis. Section 4 presents the hybrid algorithm and the corresponding convergence analysis. The fixed-gain and variable-gain algorithms are compared on the same numerical example. Application of the variable-gain algorithm to the optimization of a simulated industrial batch acrylamide copolymerization is discussed in section 5. Finally, conclusions are provided in section 6.

## 2. Run-to-Run Control

This section first shows how a finite-time dynamic process can be viewed as a static map relating input parameters available before the run and output variables measured at the end of the run. Then, the concept of run-to-run control for static processes is briefly introduced.

**Reformulation of a Finite-Time Dynamic System as a Static Map.** Consider the following finite-time dynamic system:

$$\dot{x}(t) = F(x(t), u(t)), x(0) = x_0 \quad (1)$$

$$z = h(x(t_f)) \quad (2)$$

where  $x(t)$  is the  $n$ -dimensional state vector,  $u(t)$  is the  $m$ -dimensional input vector,  $F$  is the system equations,  $x_0$  is the initial conditions,  $z$  is the  $q$ -dimensional run-end output vector,  $h$  is the output equations, and  $t_f$  is the final time of the dynamic system.

Let the infinite-dimensional input  $u(t)$  be parametrized using a finite number of parameters  $p \in \mathcal{R}^m$ , with which the inputs can be expressed as  $u(t) = \mathcal{U}(p)$ .<sup>16</sup> The final states can also be expressed in terms of the input parameters  $p$  as follows:

$$x(t_f) = x_0 + \int_0^{t_f} F(x, \mathcal{U}(p)) dt = \mathcal{F}(p) \quad (3)$$

which generates a static map between the input parameters  $p$  and the run-end outputs  $z$ , as shown in Figure 1:<sup>15</sup>

$$z(p) = h(\mathcal{F}(p)) = \mathcal{H}(p) \quad (4)$$

### Run-to-Run Control of a Finite-Time Dynamic Process.

Consider the control of a repetitive finite-time dynamic process that is characterized by two independent time variables, the run time  $t$ ,  $t \in [0, t_f]$ , and the run index  $k$ ,  $k = 1, 2, \dots$ . With run-to-run control, the repetitive nature of batch processes is exploited, whereby relevant information from the previous batches is used for computing the inputs of the subsequent run. Contrary to online control, where the input profiles are adjusted

online using run-time measurements, the idea here is to compute the input profiles of the  $(k + 1)$  run from the input profiles of the  $k$ th run and the corresponding run-end measurements.

Note that the number of manipulated inputs  $m$  has to be greater or equal to the number of measured outputs  $q$ . In the case where  $m > q$ , input decoupling<sup>4</sup> is typically performed to formulate a  $q \times q$  square control problem. As input decoupling is not the focus of this article, a square system will be assumed for simplicity, that is,  $m = q$ . Figure 2 represents schematically the run-to-run adaptation of the input parameters based on feedback, for which the inputs  $p_k$ , and thus the profiles  $u_k[0, t_f]$ , are iteratively updated such that  $z_k \rightarrow z_{\text{ref}}$  as  $k \rightarrow \infty$ . With the multivariable proportional controller  $K$ , the run-to-run control law reads

$$p_{k+1} = p_k - K(z_{\text{ref}} - z_k) \quad (5)$$

which contains integral action. The run delay seen in Figure 2 amounts to a complete batch operation. The run-to-run control scheme is described algorithmically as follows:

1. Parameterize the input profiles,  $u_k[0, t_f] = \mathcal{U}(p_k)$ .
2. Start with  $k = 1$  and  $p_k = p_1$ .
3. Implement the  $k$ th input profiles  $u_k[0, t_f]$  open loop and measure the run-end outputs  $z_k$ .
4. Determine the difference between the measured and the reference run-end outputs and compute  $p_{k+1}$  according to eq 5.
5. Set  $k = k + 1$  and return to step 3. Repeat until  $z_k = z_{\text{ref}}$  to some predefined accuracy.

Note that run-to-run control can also be used for process optimization. Indeed, if the references to follow correspond to the process optimality conditions, optimality can be achieved via run-to-run NCO tracking.

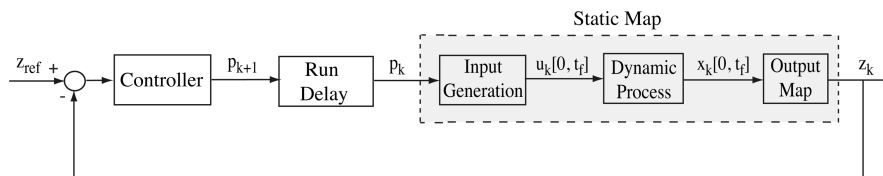
## 3. Fixed-Gain Run-to-Run Control Algorithm for Systems with Sector Nonlinearities

This section describes the concept of sector nonlinearity, presents a fixed-gain algorithm for performing run-to-run control, and analyzes its convergence property. It will be shown that the assumption of sector nonlinearities is sufficient to guarantee global convergence.

**Sector Nonlinearity.** A class of nonlinearities with agreement between the local (linear) and global pictures is considered. It is assumed that there exists a full-rank  $m \times m$  matrix  $\bar{H}$  and a scalar  $\alpha > 0$  such that

$$z^T z < \alpha z^T \bar{H}(p - p^*), \quad \forall p \neq p^* \quad (6)$$

Note that this assumption implies that there exists a  $p^*$  for which  $z = 0$ . Using the notation  $\Delta p = p - p^*$ , the classical definition of sector nonlinearity is  $(z - \zeta \bar{H} \Delta p)^T (\alpha \bar{H} \Delta p - z) > 0$ , with  $0 \leq \zeta \leq \alpha$ .<sup>10</sup> This means that the nonlinear function lies between the two linear functions  $\zeta \bar{H} \Delta p$  and  $\alpha \bar{H} \Delta p$ , as shown in Figure 3. Condition 6 is a special case of sector nonlinearity with  $\zeta = 0$  and  $\alpha > 0$ . Note that  $z(p)$  can be of any nature, for example discontinuous, within the defined sector.



**Figure 2.** Run-to-run control of a finite-time dynamic process.

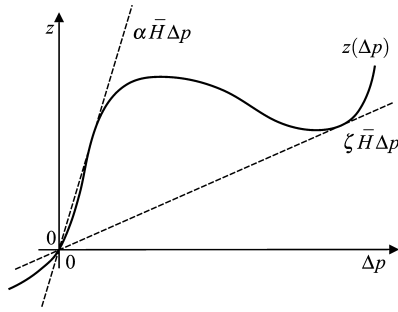


Figure 3. One-dimensional example of sector nonlinearity.

**Fixed-Gain Algorithm.** Consider the run-to-run control scheme of Figure 2. The following adaptive control law can be used, which is derived from the standard Newton–Raphson method:

$$p_{k+1} = p_k - \gamma J_k^{-1} z_k \quad (7)$$

where  $J_k = (\partial z / \partial p)|_{p_k}$  is the Jacobian of  $z(p)$  at  $p = p_k$ ,  $z_k = z(p_k)$ , and  $\gamma$  is the scalar gain of the update law.

It can be shown that, for  $\gamma = 1$  and the initialization point sufficiently close to the minimum, the Newton–Raphson method converges quadratically.<sup>13</sup> Unfortunately, if applied without any additional caution, this method has two main drawbacks:

1. It assumes the knowledge of the Jacobian  $J_k$  at all iterates  $p_k$ . However, it often happens that the expression  $z(p)$  is not known, as only the values of  $z_k$  are available, for example, via run-end measurements or through numerical integration of a dynamic system.

2. There can be points  $p_k$  for which the Jacobian  $J_k$  loses rank, for example, for the minima of  $z(p)$ . These points can make the algorithm diverge since the inverse of the Jacobian is not defined.

To circumvent the issue of Jacobian estimation, it is proposed for systems exhibiting sector nonlinearities, to choose  $J_k = \bar{H}$ , with  $\bar{H}$  a constant full-rank  $m \times m$  matrix that satisfies eq 6. The resulting fixed-gain algorithm reads:

$$p_{k+1} = p_k - \gamma \bar{H}^{-1} z_k \quad (8)$$

**Global Convergence of Fixed-Gain Algorithm.** The convergence of the fixed-gain run-to-run controller (eq 8) is investigated next for the case of sector nonlinear systems.

**Theorem 1.** Consider  $z(p)$  with  $z(p^*) = 0$ . Let  $\bar{H}$  be a full-rank  $m \times m$  matrix and  $\alpha$  a strictly positive scalar such that  $z^T z < \alpha z^T \bar{H} (p - p^*)$ ,  $\forall p \neq p^*$ . Also, consider the update law  $p_{k+1} = p_k - \gamma \bar{H}^{-1} z_k$ , with the scalar gain  $\gamma$ . Then, for  $0 < \gamma < 2/\alpha$ ,  $p_k \rightarrow p^*$  and  $z_k \rightarrow 0$  as  $k \rightarrow \infty$ .

**Proof.** The proof is based on Lyapunov direct method<sup>10</sup> and uses the Lyapunov function candidate  $V_k := V(p_k) = \Delta p_k^T \bar{H}^{-1} \bar{H} \Delta p_k$  with  $\Delta p_k = p_k - p^*$ . The fact that  $V_k > 0$ ,  $\forall p_k \neq p^*$ , and  $V(p^*) = 0$  follows from  $V$  being quadratic and  $\bar{H}$  full rank.

It will be verified that  $V_{k+1} < V_k$ ,  $\forall p_k \neq p^*$  and  $V_{k+1} = V_k$  for  $p_k = p^*$ . For this, the update law (eq 8) is rewritten as

$$\Delta p_{k+1} = \Delta p_k - \gamma \bar{H}^{-1} z_k \quad (9)$$

with  $\Delta p_k = p_k - p^*$ .

Consider first the case  $\Delta p_k = 0$ , that is,  $p_k = p^*$ . By definition of  $p^*$ ,  $z_k = 0$  and  $\Delta p_{k+1} = \Delta p_k = 0$ . Hence,  $V_{k+1} = V_k = 0$ .

Consider now the case  $\Delta p_k \neq 0$ . Equation 9 can be expressed as

$$z_k = -\frac{1}{\gamma} \bar{H} (\Delta p_{k+1} - \Delta p_k) \quad (10)$$

which, substituted into eq 6 with  $z = z_k$  and  $p = p_k$ , gives

$$\Delta p_{k+1}^T M \Delta p_{k+1} + (\alpha \gamma - 2) \Delta p_{k+1}^T M \Delta p_k + (1 - \alpha \gamma) \Delta p_k^T M \Delta p_k < 0 \quad (11)$$

with  $M = \bar{H}^T \bar{H}$ .

Since  $\Delta p_{k+1}^T M \Delta p_k = \Delta p_{k+1}^T \bar{H}^T \bar{H} \Delta p_k$  is the scalar product of  $\bar{H} \Delta p_{k+1}$  and  $\bar{H} \Delta p_k$ , the Cauchy–Schwartz inequality allows writing<sup>13</sup>

$$\Delta p_{k+1}^T M \Delta p_{k+1} \geq \frac{(\Delta p_{k+1}^T M \Delta p_k)^2}{\Delta p_k^T M \Delta p_k} \quad (12)$$

Substituting  $\Delta p_{k+1}^T M \Delta p_{k+1}$  in eq 11 by its lower bound (eq 12) and rearranging gives

$$(\Delta p_{k+1}^T M \Delta p_k + (\alpha \gamma - 1) \Delta p_k^T M \Delta p_k) (\Delta p_{k+1}^T M \Delta p_k - \Delta p_k^T M \Delta p_k) < 0 \quad (13)$$

Since the difference between the two factors in eq 13 is

$$\alpha \gamma \Delta p_k^T M \Delta p_k > 0$$

the first factor of eq 13 has to be positive and the second factor has to be negative for the inequality to be verified. It follows that

$$\frac{\Delta p_{k+1}^T M \Delta p_k}{\Delta p_k^T M \Delta p_k} > (1 - \alpha \gamma)$$

and

$$\frac{\Delta p_{k+1}^T M \Delta p_k}{\Delta p_k^T M \Delta p_k} < 1$$

or

$$(1 - \alpha \gamma) < \frac{\Delta p_{k+1}^T M \Delta p_k}{\Delta p_k^T M \Delta p_k} < 1 \quad (14)$$

From  $0 < \gamma < 2/\alpha$ , it follows that  $(2 - \alpha \gamma) > 0$ . Then, rearranging eq 11 and using  $\Delta p_{k+1}^T M \Delta p_k < \Delta p_k^T M \Delta p_k$  gives

$$\begin{aligned} \Delta p_{k+1}^T M \Delta p_{k+1} &< (\alpha \gamma - 1) \Delta p_k^T M \Delta p_k + (2 - \alpha \gamma) \Delta p_{k+1}^T M \Delta p_k \\ &< \Delta p_k^T M \Delta p_k \end{aligned} \quad (15)$$

Hence, it follows that  $V_{k+1} < V_k$  for all  $\Delta p_k \neq 0$ .

**Remarks:** (1) It is interesting to note that the Lyapunov function used in the proof is  $V_k = \bar{z}_k^T \bar{z}_k$ , with  $\bar{z}_k$  being the linear estimate of  $z_k$ , that is,  $\bar{z}_k = \bar{H} \Delta p_k$ . (2) The main difficulty with this algorithm lies in its potentially slow rate of convergence. Indeed, a large sector (large  $\alpha$ ) calls for a small gain (small  $\gamma$ , slow convergence). Hence,  $\alpha$  should be chosen as the smallest value that still satisfies eq 6. Even with this choice, the rate of convergence may be extremely slow.

#### 4. Hybrid Run-to-Run Control Algorithm for Systems with Sector Nonlinearities

As seen in the previous section, the similarity between solving sets of nonlinear equations and run-to-run control can be

exploited to generate a run-to-run controller derived from the Newton–Raphson’s most simple expression. In the context of sector nonlinear systems, using an appropriate fixed estimate of the Jacobian allows finding an upper bound for the controller gain that ensures global convergence. However, convergence can be rather slow, which, in the case of run-to-run control, can be very penalizing. In this section, it is proposed to push the use of the aforementioned similarity further. An hybrid run-to-run control algorithm, which combines variable-gain and fixed-gain updates, will be proposed. The main idea is to update the estimate of the Jacobian as long as there is no risk of being stuck in a local minimum, and to switch back to an inner loop using the fixed-gain algorithm discussed in the previous section, whenever this risk is high. Global convergence of the hybrid algorithm is proven.

As oscillations will typically occur around the solution, the Jacobian update is not only useful to increase the rate of convergence, but also to limit the oscillations around the solution by taking advantage of the quadratic convergence of quasi-Newton algorithms.

**Hybrid Algorithm.** To improve the rate of convergence, it is proposed here, instead of choosing  $J_k$  as the constant, to update the estimate of the Jacobian as in eq 7.

Let  $B_k$  denotes this estimate at the  $k$ th iterate,  $\delta_k = z_{k+1} - z_k$  and  $s_k = p_{k+1} - p_k$ . Assuming that the solution is reached at the next iteration, that is,  $z_{k+1} = 0$ , the following secant equation can be written from eq 7:

$$B_{k+1}s_k = \delta_k \quad (16)$$

Except for the case of a single nonlinear equation, eq 16 has an infinite number of solutions for  $B_{k+1}$ . Hence, the challenge is to pick a solution with good properties. Broyden’s method<sup>17</sup> updates the matrix  $B_k$  at each iteration so that the new estimate satisfies the secant eq 17. Given an initial matrix  $B_0$  (often a finite-difference approximation to the Jacobian matrix), Broyden’s method generates subsequent matrices using the update formula:

$$B_{k+1} = B_k + \frac{(\delta_k - B_k s_k) s_k^T}{s_k^T s_k} \quad (17)$$

Other formulas, for example the BFGS formula, could also be used for this update.<sup>11,14,18</sup> However, as the focus of this paper is to investigate the benefits of adding a fast outer loop to the globally convergent but slow fixed-gain controller, we will not discuss the various update laws that could be used as an alternative to Broyden’s method. Furthermore, in the context of this study with each function evaluation corresponding to a complete batch operation, it is preferred that the Jacobian update does not call for extra runs. For instance, the standard line search step, although very appealing, could lead to a significant increase in the number of runs.<sup>19</sup>

The hybrid algorithm combines variable-gain and fixed-gain updates. The variable-gain update obeys eq 7, with  $\gamma = 1$ ,  $J_k = B_k$ , and  $B_k$  updated according to eq 17, while the fixed-gain update is given by eq 8. Let us define the function  $f(p_k) = z_k^T z_k$ . The algorithm proceeds as follows: the variable-gain update is used as long as  $f(p_k)$  decreases sufficiently. Note that  $f$  will not be used thereafter as a function to minimize but rather as a test function to determine whether it is useful to switch from a variable-gain to a fixed-gain update. In other words, the algorithm proceeds with gain adaptation as long as

$$\left| \frac{f(p_{k+1}) - f(p_k)}{f(p_k)} \right| > \beta \quad (18)$$

where  $\beta$ ,  $0 < \beta < 1$ , is a scalar that is introduced to guarantee a certain reduction in  $f$  at each iteration. If the descent rate is smaller than  $\beta$ , and the solution has not been reached yet as indicated by  $f(p_k) > \varepsilon$ , with  $\varepsilon$  a small positive scalar, then a local minimum of  $z(p)$  is being approached. In such a case, the algorithm switches to the fixed low-gain update until  $f(p_k)$  has decreased sufficiently, upon which the algorithm switches back to the variable-gain update. The fact that  $f$  will decrease sufficiently using the fixed-gain update is guaranteed by Theorem 1. In fact,  $f$  can be seen as a Lyapunov function, that is, a measure of global convergence.

Solving the set of equations  $z(p) = 0$  through minimization of  $f = z(p)^T z(p)$  is a very standard approach, for which many techniques have been proposed. The two main differences between these techniques and what is proposed thereafter regard the assumption made regarding  $f$  and the convergence property. When  $f$  is used explicitly as the function to be minimized using gradient-based algorithms, global convergence to some stationary point can be proven. Most of these investigations assume that  $f$  is twice-continuously differentiable,<sup>20,21</sup> or that  $z(p)$  is Lipschitz continuous.<sup>22</sup> Here, the continuity of  $z$  is not required, and global convergence will be established such that  $z(p) \rightarrow 0$ , regardless of the initialization point.

The proposed algorithm is described in Figure 4, and its convergence will be investigated in the next subsection.  $B_0$  is initialized as  $\bar{H}$  unless a better guess is available. Furthermore, the iterations performed using the fixed low-gain update can be considered as a single step during which  $p$  and  $z$  change by  $\delta_k$  and  $s_k$ , respectively. From these variations, a Jacobian update can be performed using eq 17.

**Global Convergence of Hybrid Algorithm.** Global convergence of the hybrid algorithm is proven next.

**Theorem 2.** Consider  $z(p)$  with  $z(p^*) = 0$ . Let  $\bar{H}$  be a full-rank  $m \times m$  matrix and  $\alpha$  be a strictly positive scalar such that  $z^T z < \alpha z^T \bar{H} (p - p^*)$ ,  $\forall p \neq p^*$ . Then, the algorithm given in Figure 4 exhibits global convergence to  $z = 0$  for  $0 < \gamma < 2/\alpha$ . Moreover,  $f(p_k^0) = z_k^{0T} z_k^0$  acts as a Lyapunov function.

**Proof.** For  $f(p_k^0) = z_k^{0T} z_k^0$  to be a Lyapunov function, it needs to be positive definite and decrease with  $k$ ;  $f$  is positive definite since it is a quadratic function. Also,  $f(p^*) = 0$ . To prove that  $f(p_{k+1}^0) < f(p_k^0)$ ,  $\forall k$ , two cases need to be distinguished:

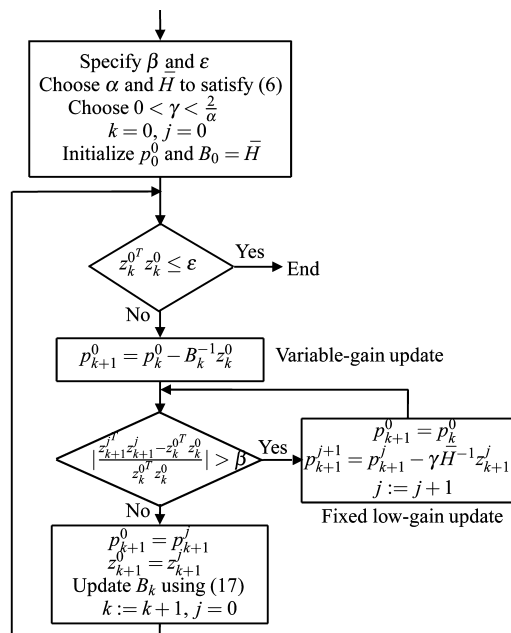
(1) If the variable-gain update generates a point for which eq 18 is satisfied, then  $f(p_{k+1}^0) < (1 - \beta)f(p_k^0) < f(p_k^0)$ .

(2) Otherwise, the algorithm switches to the fixed low-gain update. For this case, it can be shown by contradiction that  $f(p_{k+1}^0) < f(p_k^0)$ . Indeed, assume  $f(p_{k+1}^0) \geq (1 - \beta)f(p_k^0)$ . At the exit of the inner loop,  $p_{k+1}^0$  is set equal to  $p_{k+1}^i$ . Also, because of the decision to enter the  $(k + 1)$  iteration,  $f(p_k^0) = z_k^{0T} z_k^0 > \varepsilon > 0$ . Hence,  $f(p_{k+1}^i) \geq (1 - \beta)\varepsilon$ . On the other hand, since the fixed-gain update is asymptotically convergent, there always exists a  $j$  for which  $f(p_{k+1}^j) < \sigma$ , for any nonzero positive  $\sigma$ . This leads to a contradiction and, thus, it can be stated that  $f(p_{k+1}^0) < (1 - \beta)f(p_k^0) < f(p_k^0)$ .

Hence, the condition 18 must be satisfied for all  $k$ , and  $f(p_k^0)$  is a Lyapunov function for the variable-gain update that converges exponentially toward 0.

**Remarks:** (1) The same way  $V_k = z_k^{iT} z_k^i$  is a Lyapunov function for the inner loop,  $f(p_k^0) = z_k^{0T} z_k^0$  is a Lyapunov function for the outer loop. (2) Since  $f(p_k^0) = z_k^{0T} z_k^0$  is a Lyapunov function for the variable-gain update but not for the fixed-gain update,  $z_k^{iT} z_k^i$  can grow from one iteration to the next in the inner loop. (3) If  $B_k = H_k$ , the variable-gain update will be in the descent direction





**Figure 4.** Hybrid algorithm that combines variable-gain and fixed low-gain updates. The variables and functions are noted  $p_k$  and  $z_k = z(p_k)$ , respectively, with  $k$  being the iteration number of the outer loop and  $j$  being the iteration number of the inner loop. The update of  $B_k$  uses  $\delta_k = z_{k+1}^0 - z_k^0$  and  $s_k = p_{k+1}^0 - p_k^0$ .

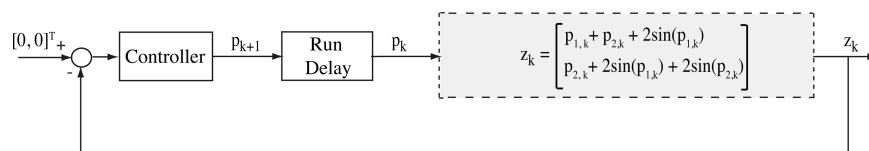
$\partial f_k = \partial f / \partial p | p_k \partial p = -z_k^T H_k B_k^{-1} z_k = -z_k^T z_k < 0$ . (4) If many calls to the inner loop are needed, it might well happen that the modified algorithm converges slower than the fixed-gain update. This means that there are many points where a full Newton step does not decrease  $f$ . In practice, this would correspond to problems that are very difficult to solve and for which a fixed low-gain update would be a robust solution. (5) In the proposed algorithm, the fixed-gain update replaces standard line-search procedures. In other words, instead of trying to satisfy the Goldstein conditions<sup>23</sup> that are based on the gradient of the objective function, “small” steps are done until the local minimum that is approached is jumped over.

**Numerical Example.** This section presents a simple numerical example, which will be used to compare the performances of the fixed-gain and variable-gain (hybrid) algorithms. Consider the  $2 \times 2$  control problem, where a static system is controlled such that  $z = [z_1, z_2]^T$  needs to be iteratively driven to its target value  $[0, 0]^T$ , by manipulating  $p = [p_1, p_2]^T$ .

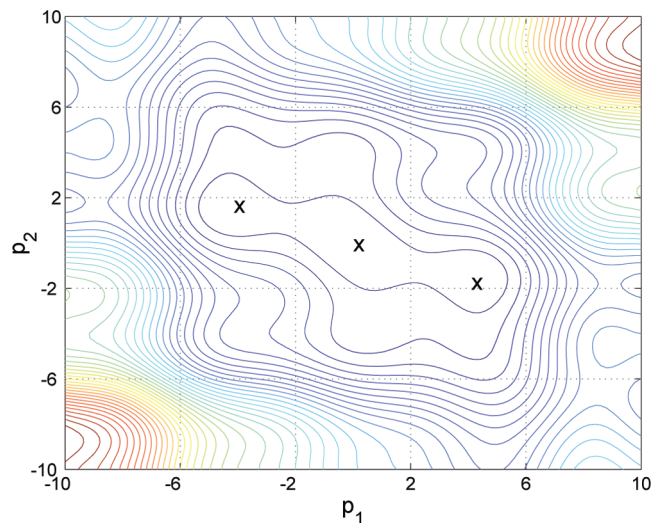
The static system and its controller are depicted in Figure 5.

A sector nonlinearity with  $\zeta = 0$  and  $\alpha = 4.67$  can be defined for this system. It follows that the maximum gain for the fixed-gain update is  $\gamma_{\max} = 2/4.67 = 0.428$ . Using the controller  $K = \gamma I_{2 \times 2}$ , with  $0 < \gamma < \gamma_{\max}$ , results in global convergence of  $z_k \rightarrow [0, 0]^T$ , regardless of the initial point  $p_0$ . As convergence can be slow, the benefits of using the variable-gain controller are investigated next.

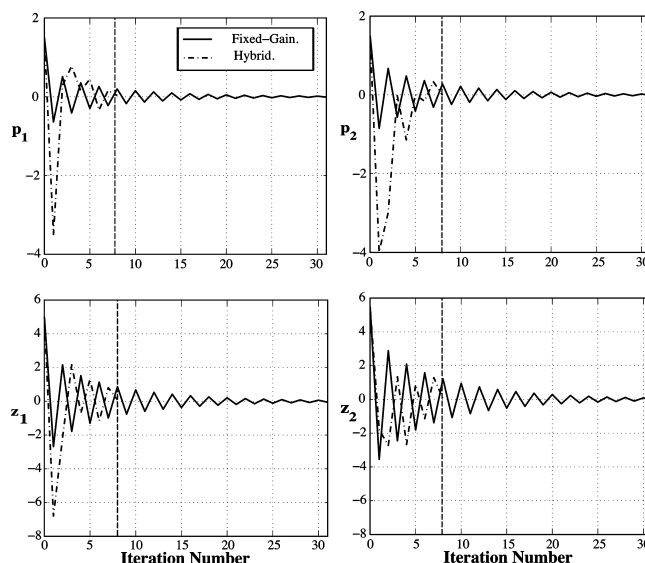
The static system is such that  $f = z^T z$  has at least three minima, two local ones and the global one  $p^* = [0, 0]^T \forall p_1, p_2 \in [-10, 10]$  as shown in Figure 6. Hence, the use of gradient-based algorithms for the minimization of  $f$ , as a way to solve



**Figure 5.** Run-to-run control of the numerical example.



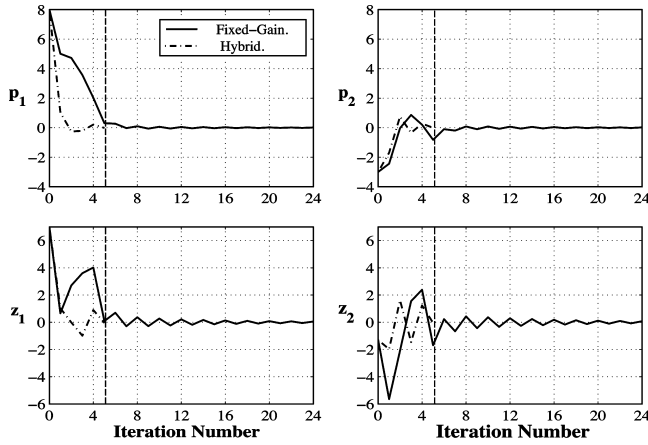
**Figure 6.** Contour of the objective function  $f = z^T z$  with the location of three minima.



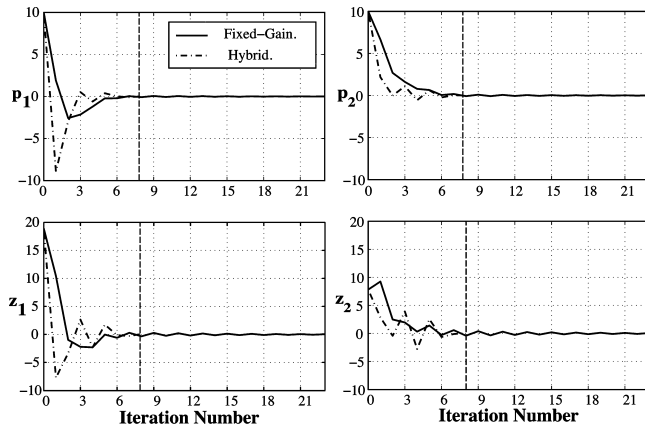
**Figure 7.** Evolution of  $z$  and  $p$  with the fixed-gain and variable-gain updates for  $p_0^0 = [1.5, 1.5]^T$ . The vertical dashed line indicates the iteration number at which the variable-gain algorithm has converged.

$z(p) = 0$ , typically leads to one of the three solutions, depending on the initialization point  $p_0$ . To ensure reaching the true solution, a global optimization algorithm must be used.

Figure 7 from Figure 9 illustrates the convergence using both the fixed-gain and hybrid updates from three different initialization points. Both approaches converge to the global solution.  $\beta$  and  $\epsilon$  are fixed to 0.01, which means that the algorithm switches to the fixed low-gain update whenever the outer loop does not succeed in reducing the function  $f$  by at least 1% between successive iterations. Table 1 shows the converged values with the two algorithms and the corresponding number of function evaluations needed to achieve convergence. With the specified values of  $\beta$  and  $\epsilon$ , only few calls to the inner loop are necessary.



**Figure 8.** Evolution of  $z$  and  $p$  with the fixed-gain and variable-gain updates for  $p_0 = [8, -3]^T$ . The vertical dashed line indicates the iteration number at which the variable-gain algorithm has converged.



**Figure 9.** Evolution of  $z$  and  $p$  with the fixed-gain and variable-gain updates for  $p_0 = [10, 10]^T$ . The vertical dashed line indicates the iteration number at which the variable-gain algorithm has converged.

As a result, the proposed hybrid algorithm is between three and six times faster than the fixed low-gain update.

**Remarks:** (1) Figure 7 to Figure 9 show that the hybrid algorithm reaches the neighborhood of the solution faster than the fixed low-gain algorithm. (2) Upon reaching the solution, the fixed-gain algorithm can oscillate around  $p^*$  as observed in Figure 7, thus penalizing the convergence rate. On the other hand, the quadratic convergence of quasi-Newton algorithms prevents these oscillations in the neighborhood of the solution, thus resulting in an even larger reduction in the number of iterations.

## 5. Case Study—Run-to-Run Control for Optimizing a Batch Polymerization Process

This section discusses the dynamic optimization of a simulated industrial batch inverse-emulsion copolymerization reactor. To keep this description as concise as possible, we will focus on the reformulated static optimization problem. A comprehensive presentation of the model is available elsewhere.<sup>7</sup>

**Dynamic Process Model.** The inverse-emulsion copolymerization of acrylamide and quaternary ammonium cationic monomers, a heterogeneous water-in-oil polymerization process, is considered. A seventh-order simplified model has been developed using industrial data:<sup>7</sup>

$$\begin{cases} \frac{dM_{1w}}{dt} = -k_{p11}R_{1T}^* \left( M_{1w} + \left( \frac{M_{2w}}{r_1} \right) \right) \\ \frac{dM_{2w}}{dt} = -k_{p12}R_{1T}^* \left( M_{2w} + r_2 \left( \frac{M_{2w}}{M_{1w}} \right)^2 \right) \\ \frac{dQ_0}{dt} = X_d Y_{d_0} \\ \frac{dQ_1}{dt} = X_d Y_{d_1} \\ \frac{dQ_2}{dt} = X_d Y_{d_2} \\ \frac{df_i}{dt} = k_{f_i}(f_{i_0} - f_i) \\ \frac{dCTA}{dt} = -k_{CTA}CTA(R_{1T}^* + R_{2T}^*), \end{cases} \quad (19)$$

where  $M_{1w}$  and  $M_{2w}$  are the concentrations of the two comonomers in the aqueous phase,  $Q_0$ ,  $Q_1$ ,  $Q_2$  are the zeroth-, first-, and second-order moments of the molecular weight distribution,  $f_i$  is the initiator efficiency, CTA is the concentration of the chain transfer agent.  $R_{iT}^*$  is the total concentration of radicals with terminal monomer groups of type  $i$  in the aqueous phase. In addition, the following quantities are used:

$$\begin{aligned} Y_{d_0} &= R_{1T}^* \\ X_d &= (k_{fm}(M_{1w} + M_{2w}) + R_{1T}^*k_{id,group} + k_{fe,group} + k_{cta}CTA) \\ Y_{d_1} &= \frac{(R_i + k_{fm}(M_{1w} + M_{2w})Y_{d_0} + k_{p,group}(M_{1w} + M_{2w})Y_{d_0})}{X_d} \\ Y_{d_2} &= \frac{(R_i + 2k_{p,group}(M_{1w} + M_{2w})Y_{d_1} + k_{fm}(M_{1w} + M_{2w})Y_{d_0})}{X_d} \end{aligned}$$

The other quantities, mainly rate constants, are described elsewhere.<sup>24,25</sup>

**Optimization Problem Formulation.** The operation is discontinuous and characterized by the repetition of several batches. There is a considerable amount of uncertainty in the form of plant-model mismatch and unmeasured process disturbances. The objective is to optimize productivity by adjusting the temperature profile between the beginning and the end of the reaction. Until very recently, the industrial practice has been to operate isothermally, with the temperature level chosen with the following in mind:

**Heat removal limitation:** Due to the exothermicity of the reactions, the reactor temperature is such that the minimal cooling temperature in the jacket approaches its lower bound. Hence, a higher reactor temperature would lead to an undesirable runaway situation.

**Molecular weight specification:** The reactor temperature is chosen to meet the molecular weight specifications. A higher

**Table 1.** Converged Values and Required Number of Iterations for Three Different Initialization Points:  $p_0^a = [1.5, 1.5]^T$ ,  $p_0^b = [8, -3]^T$ , and  $p_0^c = [10, 10]^T$

algorithm	$p_0^a$		$p_0^b$		$p_0^c$	
	converged $p^*$	no. runs	converged $p^*$	no. runs	converged $p^*$	no. runs
fixed-gain	$[0, 0]^T$	31	$[0, 0]^T$	24	$[0, 0]^T$	23
hybrid	$[0, 0]^T$	8	$[0, 0]^T$	5	$[0, 0]^T$	8

temperature in the early part of the batch would speed up the reaction but also modify the structure and properties of the polymer. However, the reactor temperature can be increased at the end of the reaction (once conversion is above 98.5%) to boost the consumption of residual acrylamide.

To increase productivity, the following minimum-time optimization problem is formulated:

$$\min_{T(t), t_f} t_f \quad (20)$$

$$\text{s.t. } \dot{x}(t) = F(x(t), T(t)), \quad x(0) = x_0 \quad (21)$$

$$X(t_f) \geq X_c \quad (22)$$

$$\bar{M}_w(t_f) \geq \bar{M}_{w,c} \quad (23)$$

$$T_{j,in}(t) \geq T_{j,in,min} \quad (24)$$

$$T(t) \leq T_{max} \quad (25)$$

where  $t_f$  is the final time to be minimized,  $T(t)$  is the reactor temperature profile,  $x(t)$  is the 7-dimensional state vector,  $F$  is the system eqs 19, and  $x_0$  is the initial conditions. The system is subject to constraints that can be divided in two categories: (1) Terminal constraints 22 and 23.  $X_c$  is the lower bound on the final conversion  $X(t_f)$ , and  $\bar{M}_{w,c}$  is the lower bound on the final average molecular weight  $\bar{M}_w(t_f)$ . (2) Path constraints 24 and 25.  $T_{j,in,min}$  is the lower bound on the coolant temperature  $T_{j,in}(t)$  at the jacket inlet, and  $T_{max}$  is the upper bound on the reactor temperature  $T(t)$ .

The optimal reactor temperature profile is computed numerically using the simplified dynamic model. On the basis of the shape of the nominal optimal solution, the reactor temperature profile is approximated as follows (so-called semiadiabatic policy):<sup>7</sup>

**Isothermal arc:** The first part is approximated by an isothermal arc. The reactor temperature is maintained constant at the value used in practice. This arc ensures that (i) the heat removal limitation is satisfied and (ii) a copolymer with the same average molecular weight as for the isothermal policy is produced during the major part of the reaction.

**Adiabatic arc:** The second part, which expresses the intrinsic compromise between conversion and quality, is approximated by an adiabatic arc. The flow rate of the cooling fluid circulating through the jacket is set to zero. The heat generated by the reaction leads to an exponential temperature increase.

As a consequence, the temperature profile can be parametrized using only two parameters: (i) the switching time between the isothermal and adiabatic arcs,  $t_{sw}$ , and (ii) the free final time  $t_f$ . Despite its intrinsic dynamic nature, from a run-to-run perspective, the relation between the decision variables  $t_{sw}$  and  $t_f$  and the controlled variables (maximal temperature, final molecular weight, and final conversion) can be seen as a static map.<sup>26,27</sup> In other words, upon completion, each run can be seen as a map between the variables  $t_{sw}$  and  $t_f$  that characterize the temperature trajectory and the output values  $T(t_f)$ ,  $\bar{M}_w(t_f)$  and  $X(t_f)$  available at final time. Hence, the optimization problem 20–25 can be rewritten as the following static optimization problem:<sup>7</sup>

$$\begin{aligned} & \min_{t_{sw}, t_f} t_f \\ & \text{s.t. } \text{static map}\{t_{sw}, t_f\} \rightarrow \{X(t_f), T(t_f), \bar{M}_w(t_f)\} \\ & T(t_f) \leq T_{max} \\ & \bar{M}_w(t_f) \geq \bar{M}_{w,c} \\ & X(t_f) \geq X_c \end{aligned} \quad (26)$$

Since (i) the constraint on molecular weight is less restrictive than that on reactor temperature, (ii) the final time is determined upon meeting the desired conversion, and (iii) the terminal constraint on reactor temperature is active at the optimum, the optimization problem reduces to

$$\begin{aligned} & \min_{t_{sw}} t_f \\ & \text{s.t. } \text{static map}\{t_{sw}\} \rightarrow \{T(t_f)\} \\ & T(t_f) = T_{max} \end{aligned} \quad (27)$$

The concept of static map can be easily understood with this example. Although  $T(t_f)$  results from the integration of the dynamic model, it only depends on the possible heat generation once the control loop is opened. In other words,  $T(t_f)$  depends on the amount of reactants left at  $t = t_{sw}$ , that is, it is a function of  $t_{sw}$ . Hence, run-to-run control can be used to enforce  $T(t_f) = T_{max}$  as depicted in Figure 10.

**Minimal-Time Operation via Run-to-Run Control.** The batch reaction time can be minimized by adapting the switching time  $t_{sw}$  to meet the terminal constraint  $T(t_f) = T_{max}$ . This problem is equivalent to solving the algebraic nonlinear equation  $z(t_{sw}) = T(t_f) - T_{max} = 0$  for  $t_{sw}$ , which can be done by adapting the switching time  $t_{sw}$  via the proposed hybrid algorithm. To ensure global convergence, one needs to verify the assumptions of Theorems 1 and 2.

Since the final temperature decreases with increasing switching time, the static gain between the input  $t_{sw}$  and the output  $T(t_f)$  is negative. Hence, the notations  $\Delta p = t_{sw} - t_{sw}^*$  and  $z = T_{max} - T(t_f)$  will lead to a sector definition for  $z(\Delta p)$  consistent with Figure 3. Figure 11 shows that  $z(\Delta p)$  lies between two linear functions of  $\Delta p$ ,  $z = 0$  and  $z = \alpha \Delta p$ . As was done in the numerical example,  $(z(\Delta p)^T z(\Delta p)) / (z(\Delta p)^T \Delta p)$  is computed to determine  $\alpha = 12.23$ . Hence, according to Theorem 2, the hybrid algorithm will converge to the optimal solution  $t_{sw}^*$ , provided  $0 < \gamma < 2/12.23 = 0.164$ . With  $\beta$  fixed to 0.01, no call to the inner loop was necessary, and convergence was achieved after 4 runs with  $\varepsilon = 10^{-6}$ .

Simulation results for the run-to-run adaptation on a 1-ton reactor are presented in Figure 12 (reactor temperature profile) and Figure 13 (molecular weight and conversion profiles). It is seen that the run-to-run optimization reduces the switching time, which increases the reactor temperature at final time. Table 2 shows that the batch time is reduced by about 40%.

**Remarks:** (1) The temperature profile for all runs remains within bounds, that is, no constraint violation occurs. (2) Run-to-run adaptation of the semiadiabatic policy is applicable to any recipe, provided a feasible initial guess of the switching time and batch-end measurements of reactor temperature are available. (3) Run-to-run adaptation can reject variations that persist over several batches. However, since no control is available during the second arc, within-run variations cannot be handled. This can be dealt with by the introduction of a backoff on  $T_{max}$ . (4) This run-to-run adaptation has been implemented in practice. A 35% -reduction in reaction time was observed.<sup>7</sup> Compared to the 40% predicted in simulation, the difference of 5% can be explained by the introduction of a backoff on  $T_{max}$  (1.8 instead of 2). (5) Using the fixed low-gain

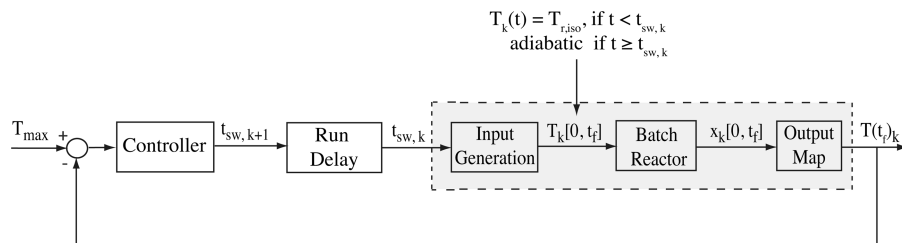


Figure 10. Run-to-run control for the optimization of the batch polymerization reactor.

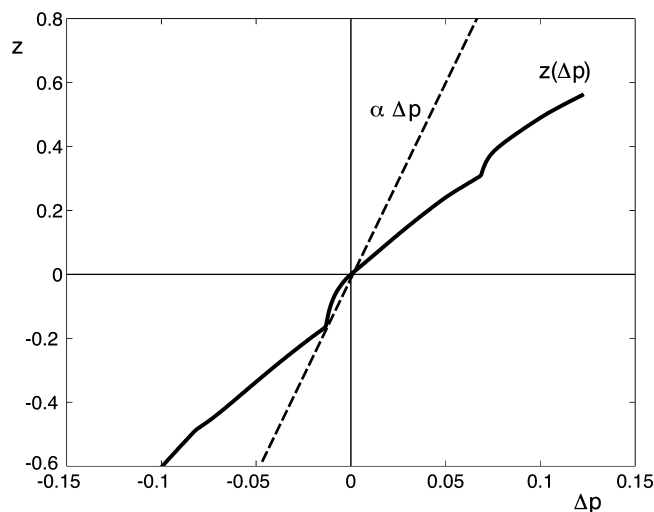


Figure 11. Sector nonlinearity for the static map between  $\Delta p = t_{sw} - t_{sw}^*$  and  $z = T_{max} - T(t_f)$ .

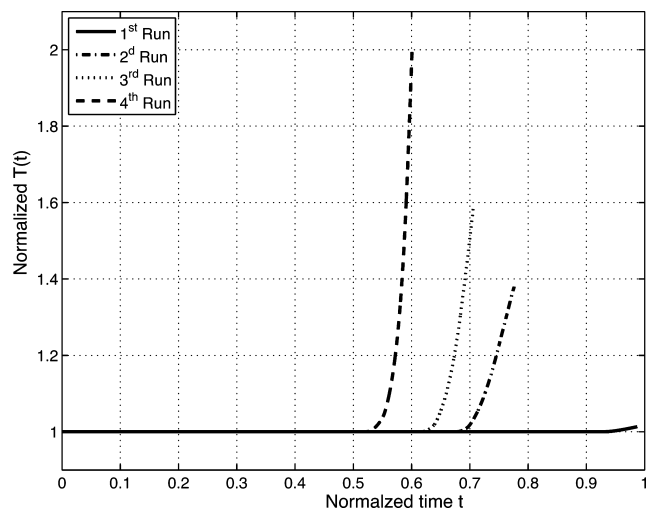


Figure 12. Simulated reactor temperature profile for switching-time adaptation using the hybrid algorithm ( $T_{max} = 2$ ).

algorithm, 15 runs are necessary to converge to the optimal solution. Hence, the hybrid algorithm is about 4 times faster, which is of considerable economic importance. This improvement is because, based on the shape of the static map between  $\Delta p$  and  $z$  (Figure 11), there is no risk of getting stuck in a local minimum. In other words, in this case, the hybrid algorithm was found as efficient as a quasi-Newton procedure, but with the guarantee of global convergence. (6) As a tendency model of the reactor was used to determine both the slope of the sector and  $t_{sw}^*$ , the following question arises naturally: why not use the value of  $t_{sw}^*$  directly on the real process? If the reactor were perfectly known, such a strategy would lead to optimality in one run. However, in the presence of uncertainty, since  $t_{sw}^*$  makes

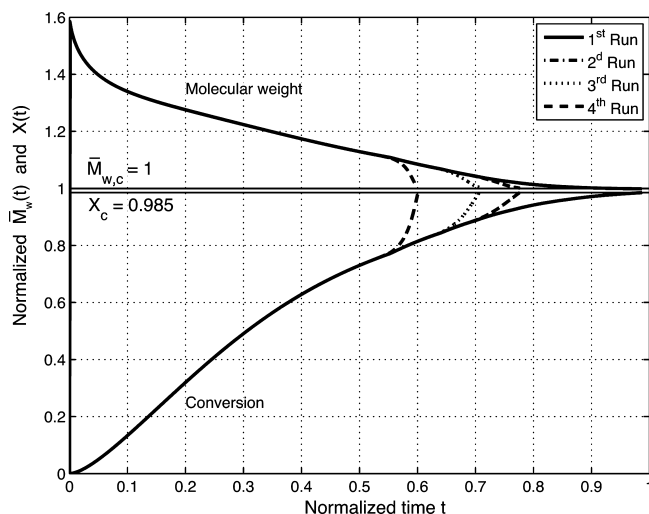


Figure 13. Simulated molecular weight and conversion profiles for switching-time adaptation using the hybrid algorithm ( $\bar{M}_{w,c} = 1$ ,  $X_c = 0.985$ ).

Table 2. Performance Improvement over 4 Runs (Objective: Reduction of  $t_f$ ; Constraint  $T(t_f) = T_{max} = 2$ )

batch	strategy	$t_{sw}$	$T(t_f)$	$t_f$
0	isothermal		1.00	1.00
1	run 1	0.928	1.01	0.987
2	run 2	0.677	1.38	0.776
3	run 3	0.616	1.59	0.706
4	run 4	0.521	1.99	0.601

the terminal constraint on  $T(t_f)$  active, there is a high risk of constraint violation, especially if the model tends to underestimate the amount of reactants that is left in the reactor at  $t_{sw}^*$ . If the model upper bounds the sector nonlinearity of the real process, then global convergence can be enforced. Conversely, in the less favorable case where the model underestimates this slope, the model could still be used to determine a robust estimate of the sector nonlinearity and thus also of the controller gain, for example via a worst-case scenario. Furthermore, the model provides an estimate of  $t_{sw}^*$  that can be used to initialize the run-to-run scheme. Note that, as long as the number of calls to the inner loop of the hybrid algorithm remains low, the performance of the hybrid scheme will not be penalized much by the conservatism on the controller gain.

## 6. Conclusions

A new run-to-run control algorithm for systems that are characterized by sector nonlinearities has been proposed. The algorithm combines variable-gain and fixed-gain updates in such a way that, through back-and-forth switching between the two types of update, (i) fast convergence can be implemented and (ii) the need for updating a Jacobian matrix close to a local minimum, where it loses rank, can be avoided. This hybrid algorithm was shown to be globally convergent. Significant improvement in convergence speed has been observed and



documented through both a simple numerical example and the run-to-run optimization of a simulated industrial polymerization process.

This hybrid algorithm has been investigated with regard to the solution of static optimization problems for which the solution is determined by active constraints. The constraints in the optimization problem represent an explicit part of the KKT conditions. Constraints satisfaction can be ensured after a certain number of runs since the proposed algorithm is globally convergent. However, to increase the rate of convergence, the inputs should be changed more aggressively, thus leading to potential constraint violation before convergence. To circumvent this difficulty, a compromise is needed between fast convergence and convergence from the feasible side of the constraints. A practical way of implementing this compromise is via appropriate choice of the gain and the introduction of backoffs from the constraints.

The main limitation of this algorithm is in regard to the difficulty of determining the slope of the upper-bounding sector. In practice, a model of the process can be used to estimate this slope. If the model is not very accurate, modeling errors can lead to poor estimates. Fortunately, the global convergence property will not be affected if the model overestimates the slope. In contrast, if the model underestimates this slope, a robust value could be determined via, for example, a worst-case scenario. This approach, though not fully satisfactory, is similar to what happens in control design, where the model is used to compute a model-based controller, which is then tuned to account for modeling errors.

A natural extension of this hybrid algorithm is the incorporation of a standard line search as an intermediate layer between the variable-gain and fixed-gain updates. If an acceptable reduction of the value of the Lyapunov function is not obtained with the variable-gain update, another point is sought in the same direction but with a smaller step size. Switching to the fixed-gain update would then only take place when the latter fails, that is, when absolutely necessary. This way, the number of calls to the fixed low-gain update, which penalizes the rate of convergence, could be minimized. Also, future research could extend the convergence analysis to other classes of nonlinear systems, for which convergence has been observed, but not proven.

## Literature Cited

- (1) Sachs, E.; Guo, R.-S.; Hu, A. Process Control System for VLSI Fabrication. *IEEE Trans. Semicond. Manuf.* **1991**, *4*, 134–144.
- (2) Wang, Y.; Gao, F.; Doyle III, F. Survey on Iterative Learning Control, Repetitive Control, and Run-to-Run Control. *J. Process Control* **2009**, *19*, 1589–1600.
- (3) Campbell, W.; Firth, S.; Toprac, A.; Edgar, T. A Survey of Run-to-Run Control Algorithms. In American Control Conference, Anchorage, 2002; pp 4212–4217.
- (4) Francois, G.; Srinivasan, B.; Bonvin, D. Use of Measurements for Enforcing the Necessary Conditions of Optimality in the Presence of Constraints and Uncertainty. *J. Process Control* **2005**, *15*, 701–712.
- (5) Castillo, E.; Yeh, J.-Y. An Adaptive Run-to-Run Optimizing Controller for Linear and Nonlinear Semiconductor Processes. *IEEE Trans. Semicond. Manuf.* **1998**, *11*, 285–295.
- (6) Bonvin, D.; Srinivasan, B.; Hunkeler, D. Control and Optimization of Batch Processes: Improvement of Process Operation in the Production of Speciality Chemicals. *IEEE Control Syst. Mag.* **2006**, *26*, 34–45.
- (7) Francois, G.; Srinivasan, B.; Bonvin, D.; Hernandez Barajas, J.; Hunkeler, D. Run-to-Run Adaptation of a Semiadiabatic Policy for the Optimization of an Industrial Batch Polymerization Process. *Ind. Eng. Chem. Res.* **2004**, *43*, 7238–7242.
- (8) Francois, G.; Srinivasan, B.; Bonvin, D. Convergence Analysis of Run-to-Run Control for a Class of Nonlinear Systems. In American Control Conference, Denver, Colorado, 2003; pp 3032–3037.
- (9) Chu, T.; Huang, L.; Wang, L. Guaranteed Absolute Stability of a Class of Delay Systems with Local Sector Nonlinearities via Piecewise Linear Lyapunov Function. In American Control Conference, Arlington, VA, 2001; pp 4212–4217.
- (10) Vidyasagar, M. *Nonlinear Systems Analysis*, 2nd ed.; Prentice Hall: Saddle River, NJ, 1993.
- (11) Dennis, J.; Schnabel, R. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*; Society for Industrial Applied Mathematics (SIAM): Philadelphia, PA, 1996.
- (12) Kelley, C. *Solving Nonlinear Equations Using Newton's Method*; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, 2003.
- (13) Kreysig, E. *Advanced Engineering Mathematics*, 6th ed.; John Wiley and Sons: New York, 1988.
- (14) Nocedal, J.; Wright, S. *Numerical Optimization*, 2nd ed.; Operations Research & Financial Engineering; Springer: NY, 2006.
- (15) Francois, G.; Srinivasan, B.; Bonvin, D. A Globally Convergent Run-to-Run Control Algorithm with Improved Rate of Convergence. In American Control Conference, Portland, Oregon, USA, 2005; pp 1901–1906.
- (16) Srinivasan, B.; Palanki, S.; Bonvin, D. Dynamic Optimization of Batch Processes: I. Characterization of the Nominal Solution. *Comput. Chem. Eng.* **2003**, *27*, 1–26.
- (17) Broyden, C. A class of Methods for Solving Nonlinear Simultaneous Equations. *Math. Comput.* **1965**, *19*, 577–593.
- (18) Bonnans, J.; Gilbert, J.; Lemaréchal, C.; Sagastizabal, C. *Numerical Optimization*; Springer: Berlin, 2003.
- (19) Fletcher, R. *Practical Methods of Optimization*; John Wiley and Sons: New York, 1991.
- (20) Gill, P.; Leonard, M. Reduced-Hessian Quasi-Newton Methods for Unconstrained Optimization. *SIAM J. Optimiz.* **2001**, 209–237.
- (21) Xiao, Y.; Wei, Z. A Modified BFGS Method Without Line Searches for Nonconvex Unconstrained Optimization. *Adv. Theor. Appl. Math.* **2006**, *1*, 149–162.
- (22) Zhou, W.; Zhang, L. Global Convergence of the Nonmonotone MBFGS Method for Nonconvex Unconstrained Minimization. *J. Comput. Appl. Math.* **2009**, *223*, 40–47.
- (23) Goldstein, A. On Steepest Descent. *SIAM J. Control* **1965**, *3*, 147–65.
- (24) Hunkeler, D.; Hamielec, A. Mechanism, Kinetics and Modelling of Inverse-Microsuspension Polymerization: 2. Copolymerization of Acrylamide with Quaternary Ammonium Cationic Monomers. *Polymer* **1991**, *32*, 2626–2640.
- (25) Hunkeler, D.; Hamielec, A.; Baade, W. Mechanism, Kinetics and Modelling of the Inverse-Microsuspension Homopolymerization of Acrylamide. *Polymer* **1989**, *30*, 127–142.
- (26) Srinivasan, B.; Primus, C. J.; Bonvin, D.; Ricker, N. L. Run-to-Run Optimization via Generalized Constraint Control. *Control Eng. Pract.* **2001**, *9*, 911–919.
- (27) Owens, C.; Zisser, H.; Jovanovic, L.; Srinivasan, B.; Bonvin, D.; Doyle, F., III. Run-to-Run Control of Blood Glucose Concentrations for People with Type 1 Diabetes Mellitus. *IEEE Trans. Biomed. Eng.* **2006**, *53*, 996–1005.

Received for review April 3, 2010

Revised manuscript received December 14, 2010

Accepted December 15, 2010

IE100808T